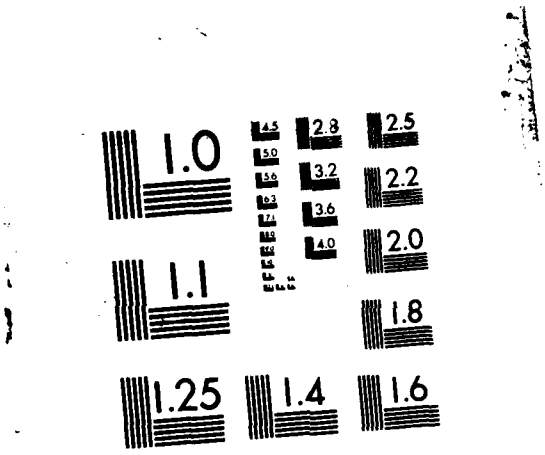
AD-A191 582 THE ADA-BASED NETWORK ARCHITECTURE SIMULATED TEST AND EVALUATION ENVIRONMENT (NASTEE/A)(U) NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA M VINEBERG JAN 88 1/1

UNCLASSIFIED F/G 12/5 NL



UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

DTIC FILE COPY

PORT DOCUMENTATION PAGE

AD-A191 582

1b. RESTRICTIVE MARKINGS

3. DISTRIBUTION/AVAILABILITY OF REPORT

Approved for public release; distribution is unlimited.

2b. DECLASSIFICATION/DOWNGRADING

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
Naval Ocean Systems Center	NOSC	Naval Ocean System. Center

6c. ADDRESS (City, State and ZIP Code)	7b. ADDRESS (City, State and ZIP Code)
San Diego, California 92152-5000	San Diego, California 92152-5000

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	AGENCY ACCESSION NO.
	63452F	EE97	AFOR	DN307 482

11. TITLE (Include Security Classification)

The Ada-Based Network Architecture Simulated Test and Evaluation Environment (NASTEE/A)

12. PERSONAL AUTHOR(S)

M. Vineberg

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Professional Paper	FROM Jun 1987 TO Jun 1987	January 1988	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	network protocols channel models message creation data transmission

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Network Architecture Simulated Test and Evaluation Environment (NASTEE), developed at the Naval Ocean Systems Center (NOSC) in San Diego, supports the characterization and validation of communication networks in general and network protocols in particular. A first generation NASTEE, written in SIMSCRIPT II.5 (NASTEE/S), was used to develop models for the LINK-11 Improvement Program, Navy MILSTAR Terminal development and VHSIC PI-Bus and TM-Bus validation. A second generation NASTEE, written in Ada (NASTEE/A), is now under development. NASTEE/A will include a library of protocols, a library of channel models and a testbed. The testbed will simulate message creation, data transmission, and damage, exercise the respective protocols and observe the network performance.

Presented at JDL Symposium on C^3 Research, Washington, DC, 16-18 June 1987.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
M. Vineberg	(619) 553-4957	Code 854

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

JDL Command and Control Research Symposium,
National Defense University, Washington, 1987

THE ADA-BASED NETWORK ARCHITECTURE SIMULATED TEST AND EVALUATION ENVIRONMENT (NASTEE/A)

Maniel Vineberg
Naval Ocean Systems Center, Code 854
San Diego CA 92152-5000

ABSTRACT

The Network Architecture Simulated Test and Evaluation Environment (NASTEE), developed at the Naval Ocean Systems Center (NOSC) in San Diego, supports the characterization and validation of communication networks in general and network protocols in particular. A first generation NASTEE, written in SIMSCRIPT II.5 (NASTEE/S), was used to develop models for the LINK-11 Improvement Program, Navy MILSTAR Terminal development and VHSIC PI-Bus and TM-Bus validation. A second generation NASTEE, written in Ada (NASTEE/A), is now under development. NASTEE/A will include a library of protocols, a library of channel models and a testbed. The testbed will simulate message creation, data transmission, and damage, exercise the respective protocols and observe the network performance.

BACKGROUND

Government and commercial experience provide ample evidence of the inherent complexity of designing communication networks: networks, which are characterized by various types of nodes, links, message types, and reliabilities, which must respond to variable stimuli (message traffic), and which are susceptible to various kinds of damage, are mathematically intractable; the structure and behavior of communication protocols, which may include acknowledgments, ignored messages, and internally generated control messages, are best represented by algorithms and finite-state machines.

To further complicate the design issue, military communication networks are components of command, control and communication (C3) systems, subject to the requirements and constraints of those systems. Changes in the C3 system requirements or advances in the technology impact the whole C3 system. Requirements for new system functions result from qualitative changes in threat; requirements for greater operating capacity result from quantitative changes in threat or available resources; advances in technology create opportunities to improve system performance.

The Naval Ocean Systems Center (NOSC) has been moving toward a methodology to model communication systems in order to (1) expose design issues early in the design process, (2) identify areas where research and development are required to alleviate technological constraints and (3) rapidly evaluate the impact of loading or design changes. The Network Architecture Simulated Test and Evaluation Environment (NASTEE) [1] is the cornerstone of this methodology.

NASTEE/S

NASTEE, implemented in SIMSCRIPT II.5 [2] to run on a VAX, is referred now to as NASTEE/S. NASTEE/S provides the following:

(1) guidelines and procedures to construct a (target) communications system model in a programming language;

(2) models of target systems, too complex to analyze mathematically, which can be tested under various configurations, loadings and assumptions, for functional correctness and performance - the models, unambiguous documentations of target systems, help to expose design errors throughout the development cycle;

(3) a library of modeling support software, communication system component models, and test scenarios, which can be drawn upon to rapidly configure and test models of new target systems;

(4) a capability to perform sensitivity analysis on previously developed models in response to proposed changes;

(5) common tools and a common approach to configuring models thereby

alleviating startup and training problems.

NASTEE/S consists of an interactive Configurer to support the input of network, loading and measurement parameters, a discrete event Testbed which executes network, loading and measurement functions in batch mode, and protocols, which are linked at run-time to provide the intelligence at each network node to process outgoing messages and incoming data.

Figure 1 illustrates how software elements are combined and tailored from the NASTEE/S library to configure new target system models.

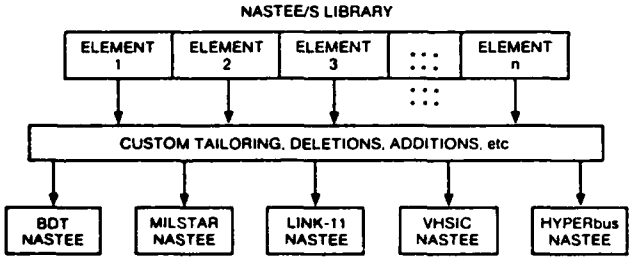


Figure 1. Separate NASTEE/S versions with common structures and origins and some common code

To conduct an experiment using NASTEE/S, a user must do the following: write the communication protocols which perform link access, routing, flow control, etc., and which detect damage and reconfigure networks; specify the experiment using the Configurer; execute the specified experiment on the Testbed. Experiment specifications include the following: switch settings to control simulation complexity and output; network configurations, including number of nodes, and node and link characteristics; operating conditions, including destruction rates, and noise sources; loading, including message types, message service, priority, and length (for each type), and message creation rates (for nodes and types); and measurement and analysis parameters to choose measures and control statistical analysis.

NASTEE/S currently supports these projects: (1) LINK-11 Improvement [3]; (2) MILSTAR terminal and baseband equipment protocols development; (3) VHSIC backplane bus validation and bus-based fault-tolerant computer architecture modeling [4]. NASTEE/S has also supported the development of a HYPERbus model (HYPERbus is a Network Systems Corporation product) [5] and the evaluation of protocols for battle-damage-tolerant shipboard networks [6].

Use of NASTEE/S has yielded several benefits: in-house experience in simulation techniques in general and NASTEE/S in specific; in-house experience in the design of certain target systems and network protocols; in-house personnel capable of supporting new target system design projects effectively with little additional training; reusable software building elements, described above, applicable to a variety of simulation projects; a program library of hardware designs and protocols, for both existing and proposed target systems, representing unambiguous documentation of those systems; techniques for comparing target systems and evaluating the costs and benefits of proposed design changes; a vehicle for evaluating proposed target system performance measures.

LESSONS LEARNED FROM NASTEE/S

Although, NASTEE/S has been used to advantage over the past two years, NASTEE/S shortcomings highlight the need for constructive change. In practice, the Configurer has been used only in support of very large network simulations. Other projects have simply drafted input specifications with a text editor; this is a problem area since the Testbed does not protect itself against input errors.

The structure of NASTEE/S - which *allows messages and frames* to act as controlling processes - has made it inconvenient to separate the protocols "cleanly" from the network descriptions. Therefore, separate versions of NASTEE/S have been created to support various target system models. This has prevented channel models and protocols developed for one target system from being used on any other system.

For each target system model, there have been attempts to decrease execution time. One approach, embedding the channel access protocol in the network description, does save execution time since nodes (or ports) with no message to send need not be active. However, this method further confuses the protocols and network description.

The mixing of protocols and network description has prevented rapid adaptation of existing protocols. Use of SIMSCRIPT II.5, not a standard for DoD, has also meant that protocols can never be transferred directly from NASTEE/S to a prototype. Also, since SIMSCRIPT II.5 is not a strongly typed language, incompatibilities are discovered at run time rather than at compile time at the expense of programmer time. The time which must be reduced is that required to develop a complete working model of a new

target system.

At a minimum, the next generation NASTEE must have these characteristics:

(1) Protocols and network descriptions must be separate;

(2) Protocols must be stored in a protocol library, must be partitioned according to a network architecture (the seven-layer ISO Model, chosen by NATO, is a strong candidate) and must conform to standard interfaces;

(3) Coding of NASTEE and the accompanying protocols must be in Ada - the new NASTEE will be NASTEE/A (A for Ada);

(4) Model development time must be reduced to 20% of the time which would be required if there were no NASTEE/A.

Separation of protocol and network descriptions and conformance to network architecture and interface standards are necessary to the construction of an independent library of protocols. The coding of NASTEE/A in Ada, a strongly typed language, alleviates the problem of run-time inconsistencies. Also, Ada generics offer the opportunity for "truly" reusable software. Protocols written in Ada may be transportable from simulation to prototype and beyond, potentially reducing system development times. The key to reducing model development time by 80% is to develop a comprehensive library of protocols and channel models.

TASS

The Ada Simulation System (TASS) was developed to enhance Ada by adding features necessary to simulation. TASS includes a library of Ada compilation units which add discrete-event simulation functions to Ada and which contain diagnostics and debugging capabilities. TASS comprises a set of Ada packages; TASS-based simulations are simply Ada programs, compiled by the Ada Compiler. No new system software is necessary.

TASS defines four object classes, ENTITY, QUEUE, PROCESS and RESOURCE, and operations related to each class. An ENTITY is an object with associated data. A node is an example ENTITY; a node identification number and the number of associated I/O ports are examples of associated data. Operations on ENTITIES include Create and Delete.

A QUEUE is an ordering of objects. A buffer is an example QUEUE; input words are example QUEUE members. Operations on QUEUEs include Create, Delete, Include (a member in the QUEUE), Remove (the member from the QUEUE), Find (a member in a QUEUE) and Delete (a member from a QUEUE).

A PROCESS is an object that performs work, such as a network interface unit. Operations on PROCESSes include Create, Delete, Schedule, Work (for a certain length of time), Suspend (this PROCESS - a PROCESS may suspend itself), Reactivate (another PROCESS - a suspended PROCESS cannot reactivate itself), Interrupt, and Resume.

A RESOURCE is an object with limited access, such as a processor. Operations on RESOURCEs include Create, Delete, Request and Relinquish. A PROCESS requesting a RESOURCE may not complete until access to that RESOURCE is granted. Access to that RESOURCE may not be granted to a second PROCESS until the first PROCESS relinquishes the RESOURCE.

NASTEE/A

The conceptual view of NASTEE/A includes several high-level components, as shown in Figure 2. The TESTBED, which will comprise a network model as well as message load and damage models, will be separate from the communications protocols. Those protocols will reside in a PROTOCOL LIBRARY (it is expected that the PROTOCOL LIBRARY will incorporate impending NATO standards). The USER will have write access to an INPUT FILE and read access to OUTPUT FILES.

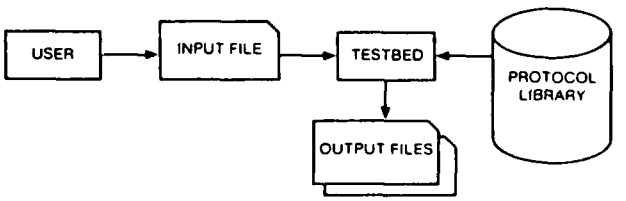


Figure 2. NASTEE/A Block Diagram

USER

The USER validates that a model accurately represents a target system specification and verifies that the model performs specified target system functions. As such, the USER must draft experiments which specify network configurations, message loads, operating conditions (such as noise and damage), and measurements.

INPUT FILE

The INPUT FILE comprises five conceptual profiles; the profiles themselves may be partitioned in the actual Ada implementation to reflect the choice of Ada packages.

1. A Net Profile specifies the nodes, ports and channels that comprise the target system. A node connects to a channel through a port. Two important attributes of a port specification are the identifier of a channel to which it is connected and the name of a protocol which it will use to process messages.

2. A Message Profile specifies the load under which the target system will be tested. It allows for the creation of both fixed-parameter and random-parameter messages (NASTEE/S requires the user to select one or the other message-creation method). The Message Profile comprises two components: a table specifying messages with fixed parameters (e.g., creation time, message type, origin node and port, destination node, length, and priority); a set of random message features (e.g., types, creation rates, length and priority distributions, origins, and destinations).

3. A Damage Profile specifies special events designed to interfere with the operation of the target system. It allows for the creation of two types of damage, permanent (destruction of a node or channel or a "stuck-at" fault) and transient (a bit error during a single cycle); either type of damage may be specified to occur under fixed parameters or at random. The Damage Profile comprises two components: a table specifying damage with fixed parameters (e.g., type, time, and object to be damaged); a set of random damage features (e.g., types and probabilities).

4. An Execution Profile specifies how a TESTBED experiment is to be conducted. The Run Time and all switch settings (e.g., mode and iebug switches) are specified here.

5. An Observation Profile specifies what data will be extracted during an experiment, how and when those data will be analyzed, and when the results will be output. Examples of data are message queue lengths, message delays, messages received, error reports, and channel usage.

TESTBED

The TESTBED consists of these models: a Node Model; a Channel Model; a Message Model; a Damage Model; an Observation Model; and an Execution Model (driver).

Figure 3 shows how these models interact with the Input and Output Files. The Node and Channel Models obtain attributes from the Net Profile. The Message and Damage Models obtain attributes from their respective profiles. The Observation Profile is input to all models (thereby affecting their data gathering). The Execution Profile is input to the Execution Model which controls the operation of all models. The Message model will affect the operation of the Node Model; the Damage Model may affect the operation of the Node and Channel Models. All results pass via the Observation Model to a Result File. There will also be diagnostic outputs to other unformatted output files. The Node Model interacts with the Protocol Library through Protocol Interfaces. All components of the block diagram are described below.

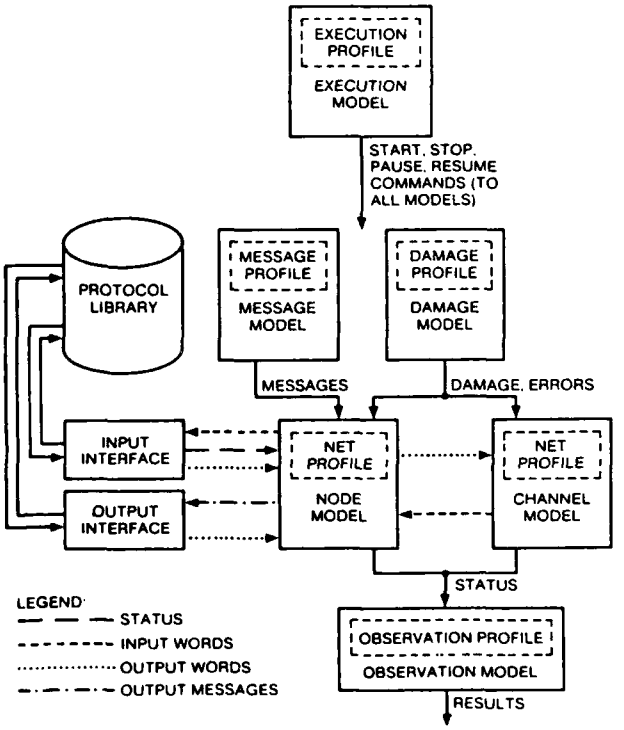


Figure 3. NASTEE/A TESTBED Block Diagram

1. The Node Model includes the logic to perform node application functions, including any required interpretation and creation of messages. The Node Model also incorporates the logic governing the operation of each port associated with each node. A node must have a port for each channel on which it communicates.

Each port is characterized by its parent node, the channel to which it is connected, its protocols (name and bottom and top layers), a protocol interface, and a channel interface.

The protocol interface consists of buffers for information required by a protocol from a port (e.g., a message to be translated) and required by the port from the protocol (e.g., transmittable units representing the translated message). The channel interface consists of input and output buffers containing "channel words" (bit strings which are transmittable during a single time unit); these channel words represent the data which have been read from or which are to be written to the channel, respectively.

The basic unit of TESTBED time is the "cycle". Each port causes its protocols to be executed continuously; the USER may specify the number of cycles required to execute various protocols. Both port and protocols have access to the respective protocol interfaces. A port puts the "next" word from its channel interface output buffer onto the channel at each cycle; then, during the same cycle, it reads a word from the channel into the channel interface input buffer. Actions at the port may be triggered by information received on the channel or by information, such as a timeout indication, in protocol interface registers.

It is expected that the Node Model will be entirely reusable.

2. The Channel Model is sensitive to the definition of a channel. An HF channel model might include a single channel, representing the air waves over which all nodes communicate. A ring channel model might include one or more channels for each pair of nodes which communicate directly. A channel model accepts an output word from each port (output from the ports is input to the channel) on each channel and then, according to the logic of the model, delivers an input word to each port on each channel.

For example, if undisturbed by the damage model, for the PI-Bus (the Parallel-Interface backplane bus specified under VHSIC) [7], the LOGICAL OR of all port output words would be delivered as the input word to each port. An HF channel model might consider platform position and velocity, antenna coverage, atmospheric noise, and jamming in synthesizing the various port input words. Once the input word is available, the port loads it into its Protocol Interface from where it is processed (or ignored) by the respective protocol.

It is expected that a set of reusable Channel Models will be required.

3. The Message Model creates messages as specified in the Message Profile and then loads these into message buffers at the respective ports of origin. Messages of various types may be created at various rates at the various ports.

It is expected that the Message Model will be entirely reusable.

4. The Damage Model creates damage as specified in the Damage Profile. A node or channel is destroyed as specified. Transient errors are introduced once; stuck-at faults are introduced at every cycle following the initial occurrence. A channel error is injected between the respective channel read and write cycles. A port error is injected prior to the respective channel write cycle or after the respective channel read cycle.

It is expected that the Damage Model will be entirely reusable.

5. Based on the Observation Profile, the above four models will issue requests to the Observation Model to record data. The Observation Model will record and, according to the Observation Profile, analyze the data. Then, as specified, the Observation Model will write the results to RESULT FILES; this may be done periodically, to record intermediate results, or once only, at the end of an experiment.

It is expected that the Observation Model will be reusable but will evolve according to user requirements.

6. The Execution Model controls the operation of the TESTBED. It comprises a driver which will initialize, start, suspend, restart, and stop the simulation, and will request output from the Observation Model as appropriate.

It is expected that the Execution Model will be entirely reusable.

PROTOCOL LIBRARY

The PROTOCOL LIBRARY, will be built according to the seven-layer ISO Model (existing protocols which cannot be expressed in the context of this model will require new interface definitions). Each protocol will be written as a set of routines, separately callable at each layer for each direction, input and output.

It is expected that new protocols will be written for each new target

system. However, the key to reducing target system modeling time will be to build a PROTOCOL LIBRARY containing reusable protocols. As the PROTOCOL LIBRARY becomes an extensive resource, it will be possible to initially configure a target system model from reusable, "okay" protocols; new, "good" protocols will be developed where required and then integrated and tested when ready. In some cases, it will be possible to develop the new protocols by modifying or enhancing existing protocols, thereby further reducing development time.

OUTPUT FILES

The OUTPUT FILES have yet to be specified. It is expected that a formatted Result File will be written from the Observation Model. Other unformatted output files will accept outputs as generated by the other TESTBED models. At least one OUTPUT FILE will serve as a graphics interface file. On the TESTBED (input) side of this file, the graphics interface file will accept standard requests for graphics service. These virtual TESTBED service requests will then be translated into specific service requests for whatever graphics system is finally chosen for NASTEE/A.

SUMMARY AND STATUS

Much has been learned from using SIMSCRIPT-based NASTEE/S. These lessons are being incorporated into the development of Ada-based NASTEE/A. When complete, NASTEE/A should facilitate more rapid development of target system models. If the protocols for these target system models are also written in Ada, the development times of the target systems themselves should be reduced.

A first version of TASS has been released and is now in use to develop NASTEE/A. An early version of NASTEE/A will be ready by the end of 1987 to support development of Ada versions of PI-Bus [7] and TM-Bus [8] models. Upon completion of those bus models, NASTEE/A will undergo evaluation, revision and subsequent release for use on more extensive models.

ACKNOWLEDGEMENTS

The author takes great pleasure in acknowledging the contributions to this paper of Robert Ollerton, Darlene Teotico, and Gary Brown, all of the Naval Ocean Systems Center.

REFERENCES

1. Vineberg, M., Norvell, S. and Keune, C., "Network Architecture Simulated Test and Evaluation Environment (NASTEE)," Proc 7th MIT/ONR Workshop on C3 Systems, San Diego, Dec 84, 109-114.

2. C.A.C.I., "SIMSCRIPT II.5 Reference Handbook," CACI, Inc.-Federal Los Angeles, Mar 85.

3. Norvell, S., Brown, G. and Vineberg, M., "Modeling and Evaluation of Multiple Access Protocols," Record of 1985 IEEE Military Communications Conference, Boston, Oct 85, 128-132.

4. Vineberg, M., "VHSIC Backplane Bus Modeling and Validation," Final Report (to the VHSIC Program Office), Naval Ocean Systems Center, Code 854, San Diego, June 1986.

5. Crossland, K. and Vineberg, M., "Modeling and Evaluation of Local Area Networks," Record of 1985 IEEE Military Communications Conference, Boston, Oct 85, 428-431.

6. Vineberg, M., "A Methodology to Test and Evaluate and Evaluate Battle Tolerance of Shipboard Networks," 1986 Digest of Papers, Government Microcircuit Applications Conference (GOMAC), San Diego, Nov 86, 251-254.

7. IBM Honeywell TRW, "VHSIC Phase 2 Interoperability Standards, PI-Bus Specification," version 2.1, 26 September 1986.

8. IBM Honeywell TRW, "VHSIC Phase 2 Interoperability Standards, TM-Bus Specification," version 2.0, 31 December 1986.

END

DATE

FILMED

5-88

DTIC